Assignment No: 03D

Title of the Assignment: Implement Greedy search algorithm.

Problem statement: Implement Selection Sort algorithm.

Objective:

- To understand the concept of greedy search algorithm and sorting algorithms.
- To implement Selection sort algorithm.

Theory:

Greedy Method/Technique:

The greedy method is one of the strategies like Divide and conquer used to solve the problems. This method is used for solving optimization problems. An optimization problem is a problem that demands either maximum or minimum results.

The Greedy method is the simplest and straightforward approach. It is not an algorithm, but it is a technique. The main function of this approach is that the decision is taken on the basis of the currently available information. Whatever the current information is present, the decision is made without worrying about the effect of the current decision in future.

This technique is basically used to determine the feasible solution that may or may not be optimal. The feasible solution is a subset that satisfies the given criteria. The optimal solution is the solution which is the best and the most favorable solution in the subset. In the case of feasible, if more than one solution satisfies the given criteria then those solutions will be considered as the feasible, whereas the optimal solution is the best solution among all the solutions.

Applications of Greedy Algorithm:

- It is used in finding the shortest path.
- It is used to find the minimum spanning tree using the prim's algorithm or the Kruskal's algorithm.
- It is used in a job sequencing with a deadline.
- This algorithm is also used to solve the fractional knapsack problem.

"A greedy algorithm is any algorithm that follows the problem-solving heuristic of making the locally optimal choice at each stage with the intent of finding a global optimum." (Kleinberg and Tardos, Algorithm Design, 2005)

Selection Sort:

A greedy algorithm is one that makes the locally optimal choice at each step with the hope of finding a global optimum. In selection sort, the algorithm selects the smallest (or largest) element from the unsorted portion of the array and places it at the beginning of the sorted portion of the array. This is done repeatedly until the entire array is sorted.

At each step, selection sort chooses the locally optimal choice (the smallest element) and places it in its correct position in the sorted portion of the array. While this choice may not always lead to the globally optimal solution, it is the best choice based on the information available at that particular step of the algorithm. Selection sort does follow the problem-solving heuristic of making the locally optimal choice at each stage, and this is why it is considered a greedy method.

Selection sort is a sorting algorithm that selects the smallest element from an unsorted list in each iteration and places that element at the beginning of the unsorted list.

Procedure for Selection Sort

1. Swap the first element with the smallest element in the array. We have now n-1 elements to sort
2. Find the smallest element in array from array[1] to array [n-1] and swap it with the second position
3. For the next element, find the smallest element in the remaining array and swap it.

Working of Selection Sort

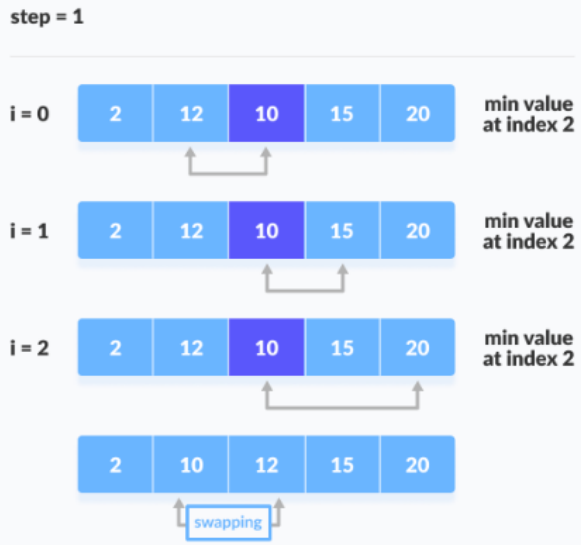
1. Set the first element as `minimum`

2. Compare `minimum` with the second element. If the second element is smaller than `minimum`, assign the second element as `minimum`.

   Compare `minimum` with the third element. Again, if the third element is smaller, then assign `minimum` to the third element otherwise do nothing. The process goes on until the last element.

3. After each iteration, `minimum` is placed in the front of the unsorted list.

4. For each iteration, indexing starts from the first unsorted element. Step 1 to 3 are repeated until all the elements are placed at their correct positions.



The second iteration

The third iteration

The fourth iteration

Selection Sort Algorithm

```
selectionSort(array, size)
  repeat (size - 1) times
  set the first unsorted element as the minimum
  for each of the unsorted elements
    if element < currentMinimum
      set element as new minimum
  swap minimum with first unsorted position
end selectionSort
```

Selection Sort Complexity:

Time Complexity	
Best	$O(n^2)$
Worst	$O(n^2)$
Average	$O(n^2)$
Space Complexity	O(1)
Stability	No

Conclusion:

We have implemented Selection sort algorithm.

Oral Questions:

1. What is selection sort and how does it work?
2. What is the time complexity of selection sort and why?
3. Explain the algorithm for selection sort step-by-step?
4. How is selection sort different from other sorting algorithms, such as bubble sort and insertion sort?
5. Can you provide an example of how selection sort can be used in a real-world application?
6. How would you modify selection sort to sort an array in descending order?
7. Is selection sort stable or unstable? Can you explain why?
8. How would you analyze the efficiency of selection sort for large input sizes?
9. Are there any limitations or disadvantages of using selection sort? If so, what are they and how can they be addressed?